**Project Development Phase**
**Model Performance Test**

| | |
|---|---|
| Date | 4th November 2023 |
| Team ID | 592358 |
| Project Name | Walmart Store Sales Forecasting |
| Maximum Marks | 10 Marks |

**Model Performance Testing:**

Project team shall fill the following information in model performance testing template.

| S.No. | Parameter | Values | Screenshot |
|---|---|---|---|
| 1. | Metrics | **Regression Model:**<br>MAE - , MSE - , RMSE - , R2 score -<br><br>**Classification Model:**<br>Confusion Matrix - , Accuray Score- & Classification Report - | |
| 2. | Tune the Model | Hyperparameter Tuning -<br>Validation Method - | |

**Regression Model:**

MAE - , MSE - , RMSE - , R2 score

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
import numpy as np

# Define your features (X) and the target variable (y)
X_rf = data.drop(columns=['Weekly_Sales','Date'])
y_rf = data['Weekly_Sales']

# Split the data into a training set (70%) and a testing set (30%)
X_train_rf, X_test_rf, y_train_rf, y_test_rf = train_test_split(X_rf, y_rf, test_size=0.3, random_state=42)

# Initialize and train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_rf, y_train_rf)

# Make predictions on the test data
y_pred_rf = rf_model.predict(X_test_rf)

# Calculate R-squared (accuracy), RMSE, and MAE
r2_rf = r2_score(y_test_rf, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test_rf, y_pred_rf))
mae_rf = mean_absolute_error(y_test_rf, y_pred_rf)

print("Random Forest Model:")
print("R-squared (Accuracy):", r2_rf)
print("Root Mean Squared Error (RMSE):", rmse_rf)
print("Mean Absolute Error (MAE):", mae_rf)
```

```
Random Forest Model:
R-squared (Accuracy): 0.9625817498533463
Root Mean Squared Error (RMSE): 4438.228933960859
Mean Absolute Error (MAE): 1648.1275767946804
```

**Classification Model:**

Confusion Matrix - , Accuray Score- & Classification Report –

## Confusion Matrix - , Accuray Score- & Classification Report -

In [41]:
```
# Define a threshold to binarize predictions (adjust as needed)
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

threshold = 20000
y_pred_rf_class = np.where(y_pred_rf > threshold, 1, 0)

# Convert actuals to binary as well for comparison
y_test_rf_class = np.where(y_test_rf > threshold, 1, 0)

# Calculate confusion matrix, accuracy score, and classification report
conf_matrix = confusion_matrix(y_test_rf_class, y_pred_rf_class)
accuracy = accuracy_score(y_test_rf_class, y_pred_rf_class)
class_report = classification_report(y_test_rf_class, y_pred_rf_class)

print("Random Forest Classification Metrics:")
print("Confusion Matrix:\n", conf_matrix)
print("Accuracy Score:", accuracy)
print("Classification Report:\n", class_report)
```

```
Random Forest Classification Metrics:
Confusion Matrix:
 [[92564  1751]
 [ 1550 30606]]
Accuracy Score: 0.973899154746938
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     94315
           1       0.95      0.95      0.95     32156

    accuracy                           0.97    126471
   macro avg       0.96      0.97      0.97    126471
weighted avg       0.97      0.97      0.97    126471
```

Hyperparameter Tuning

## Hyperparameter Tuning

In [ ]:
```
from sklearn.model_selection import RandomizedSearchCV

# Initialize the Random Forest model
rf_model = RandomForestRegressor(random_state=42)

# Define a hyperparameters grid for tuning with higher values
param_dist = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 20, 30],
}

# Create the RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=rf_model, param_distributions=param_dist, n_iter=5, cv=3, scoring='neg_mean_squared_

# Fit the model to the training data
random_search.fit(X_train_rf, y_train_rf)

# Get the best hyperparameters
best_params_random = random_search.best_params_

# Initialize and train the Random Forest model with the best hyperparameters
best_rf_model_random = RandomForestRegressor(**best_params_random, random_state=42)
best_rf_model_random.fit(X_train_rf, y_train_rf)

# Make predictions on the test data
y_pred_rf_random = best_rf_model_random.predict(X_test_rf)

# Calculate R-squared (accuracy), RMSE, and MAE for the tuned model
r2_rf_random = r2_score(y_test_rf, y_pred_rf_random)
rmse_rf_random = np.sqrt(mean_squared_error(y_test_rf, y_pred_rf_random))
mae_rf_random = mean_absolute_error(y_test_rf, y_pred_rf_random)

print("Random Forest Model (Randomized Search - Increased n_estimators and max_depth):")
print("Best Hyperparameters:", best_params_random)
print("R-squared (Accuracy):", r2_rf_random)
print("Root Mean Squared Error (RMSE):", rmse_rf_random)
print("Mean Absolute Error (MAE):", mae_rf_random)
```

Validation Method -

```
]: # Create a DataFrame with the input data
input_data = pd.DataFrame({
    'Store': [1],
    'Dept': [1],
    'IsHoliday': [0],
    'Type': [0],
    'Size': [151315],
    'Temperature': [42.31],
    'Fuel_Price': [2.572],
    'MarkDown1': [7246.420196],
    'MarkDown2': [3334.628621],
    'MarkDown3': [1439.421384],
    'MarkDown4': [3383.168256],
    'MarkDown5': [4628.975079],
    'CPI': [211.096358],
    'Unemployment': [8.106],
    'Month': [2],
    'Year': [2010],
    'Day_of_Week':[4]
})

# Use the trained random forest model to make the prediction
predicted_sales = rf_model.predict(input_data)

# Print the predicted weekly sales
print("Predicted Weekly Sales:", predicted_sales[0])
```

```
Predicted Weekly Sales: 26689.844299999997
```